

METHODS, SYSTEMS AND COMPUTER PROGRAM PRODUCTS FOR  
SECURE DELEGATION USING PUBLIC KEY AUTHENTICATION

Field of the Invention

The present invention relates to authentication and more particularly to  
5 authentication of a client when delegation is utilized to access a server.

Background of the Invention

Networked computer applications are often deployed using a "tiered"  
model. In this model, the originator of a request for a unit of work (also referred to  
10 as a "principal") typically initiates that work via a client program (first tier), which  
then communicates to a web server, or similar second tier server (also referred to as  
a middle-tier server), which itself communicates, on behalf of the request  
originator, to other middle-tier servers and/or to third or fourth tier servers such as  
database servers or other resource managers. When the request is processed by the  
15 resource managers, they, typically, evaluate whether the request originator has been  
authenticated and whether they are authorized to perform the unit of work. The  
resource managers, typically, also record access by the originator of the request in  
appropriate audit logs.

Such a tiered approach to networked applications may create a need for the  
20 secure propagation of security credentials of the request originator through each of  
the tiers of the application. In such propagation of secure credentials, the request  
originator delegates to the middle-tier servers the authority to access other servers

on their behalf. Thus, the secure propagation of the credentials of the request originator (the requesting "principal") may be referred to as "delegation" or "impersonation."

One conventional approach for asynchronous message based authentication is to create a digital signature for the message. The digital signature is based on a public/private key pair. An example of such a digital signature approach to authentication is Public Key Infrastructure (PKI) authentication. In PKI, typically, a nonce, which may, for example, be a 60 bit random number, is generated by a party, such as a server, and provided to the client. The client signs the nonce with its digital signature and returns the signed nonce to the server. Typically, the server evaluates the digital signature of the client by decrypting the signed nonce with the public key of the client, which may be obtained from a certificate associated with the client, and comparing the decrypted nonce to the nonce originally sent. If the nonces are the same, the signature is authentic. In such a manner, the server may be assured of the authenticity of the client. This manner of authenticating the client is used in a variety of computer protocols, including Secure Sockets Layer (SSL) and Transport Layer Security (TLS).

One difficulty with such a PKI authentication procedure is that it may be difficult to provided delegation of client authentication in certain circumstances. For example, a request from a principal through a client may pass through a middle-tier server which, in response to the request, accesses multiple third or fourth tier servers (also referred to as back-end servers). In such a case, the middle-tier server may need to authenticate the principal or the client to multiple back-end servers. Such a delegation of authentication may difficult in light of the multiple servers for which the client may need authentication.

### Summary of the Invention

Embodiments of the present invention provide methods, systems and computer program products for a middle-tier server to impersonate a client to a plurality of servers. A common nonce associated with each of the plurality of servers is obtained and the common nonce is sent to the client. The common nonce

is digitally signed by the client and is received at the middle-tier server and provided as a signature for transactions from the client to the plurality of servers so as to authenticate the client to the plurality of servers.

In further embodiments of the present invention, obtaining a common  
5 nonce is provided by generating a common nonce based on information obtained from each of the plurality of servers. In such embodiments, generating the common nonce may be accomplished by obtaining pre-nonce contributions from the plurality of servers, combining the pre-nonce contributions to provide a single pre-nonce token and providing the common nonce based on the pre-nonce token.  
10 The common nonce may be provided by reducing the pre-nonce token to provide the common nonce. Furthermore, the pre-nonce contributions may be combined to provide a single pre-nonce token by concatenating the pre-nonce contributions. Also, the pre-nonce token may be reduced to provide the common nonce by hashing the pre-nonce token utilizing a one-way hash function so as to provide the  
15 common nonce.

In additional embodiments of the present invention, obtaining pre-nonce contributions may be provided by requesting a pre-nonce contribution from each of the plurality of servers and receiving the pre-nonce contributions from the plurality of servers. The request for a pre-nonce contribution may be provided by sending  
20 authenticated requests to the plurality of servers. Additionally, the authenticated requests may be sent to the plurality of servers may be encrypted. The authenticated request may include at least one of an identification of a source of the request, a time stamp and a random number.

In still further embodiments of the present invention, the pre-nonce  
25 contributions include at least one of an identification of a server of the plurality of servers and a random number. Furthermore, the pre-nonce contributions may be signed with a signature corresponding to a server from which the pre-nonce contribution was obtained. In such embodiments, the signatures may be incorporated in the pre-nonce token.

30 In yet further embodiments of the present invention, the pre-nonce contributions are signed with a signature corresponding to a server from which the

pre-nonce contribution was obtained. In such embodiments, the signatures of the pre-nonce contributions are authenticated and the pre-nonce contributions for which the digital signature are not authentic are rejected.

In still further embodiments of the present invention, a transaction  
5 identification is received from a trusted server of the plurality of servers and the transaction identification associated with the common nonce. Use of the common nonce may be tracked based on the transaction identification.

In additional embodiments of the present invention, an expiration time is associated with a pre-nonce contribution and it is determined if the pre-nonce  
10 contribution has expired based on its associated expiration time. In such embodiments, the common nonce may be received at a server of the plurality of servers and a pre-nonce contribution associated with the received common nonce is determined. The received common nonce is accepted if the associated pre-nonce contribution has not expired.

15 In yet additional embodiments of the present invention, at least one of the plurality of servers receives a client certificate, determines if the client certificate is trusted and indicates that the client is not authenticated if the client certificate is not trusted. Furthermore, the signed common nonce and a client certificate may be received and it is determined if the signature of the signed common nonce  
20 corresponds to a signature of the client certificate. The client is not authenticated if the signature of the signed common nonce does not correspond to the signature of the client certificate. The signed common nonce and the pre-nonce token may also be received and the received pre-nonce token hashed. The hashed pre-nonce token is compared to the common nonce and the client is not authenticated if the hashed  
25 pre-nonce token is different from the common nonce. The pre-nonce token may also be received at one of the plurality of servers where it is determined if the pre-nonce token includes a pre-nonce contribution from the receiving server. For example, if such contributions are digitally signed before they are contributed, the receiving server may verify its signature. The client is not authenticated if the pre-  
30 nonce token does not include the random number associated with the receiving server. Additionally, an expiration may be associated with the random number

associated with the at least one of the plurality of servers and the client is not authenticated if the pre-nonce token does not include a random number associated with the at least one of the plurality of servers which has not expired.

In still further embodiments of the present invention, the common nonce is  
5 obtained by obtaining the common nonce from a party trusted by the middle-tier server and the plurality of servers. The common nonce is signed by the trusted party. The signature of the common nonce is verified the signature of the trusted party. In further embodiments, at least one of the plurality of servers receives a client certificate and determines if the client certificate is trusted The client is not  
10 authenticated if the client certificate is not trusted. The signed common nonce and a client certificate may also be received and it determined if the signature of the signed common nonce corresponds to a signature of the client certificate. The client is not authenticated if the signature of the signed common nonce does not correspond to the signature of the client certificate.

15 As will further be appreciated by those of skill in the art, while described above primarily with reference to method aspects, the present invention may be embodied as methods, apparatus/systems and/or computer program products.

#### Brief Description of the Drawings

20 **Figure 1A** is a block diagram illustrating a system incorporating embodiments of the present invention;

**Figure 1B** is a block diagram illustrating a system incorporating alternative embodiments of the present invention;

**Figure 2** is a block diagram of a data processing system according to  
25 embodiments of the present invention;

**Figure 3** is a more detailed block diagram of a data processing system according to embodiments of the present invention;

**Figure 4** is a flowchart illustrating operations of a middle-tier server according to embodiments of the present invention;

30 **Figure 5** is a flowchart illustrating operations for common nonce generation according to embodiments of the present invention;

**Figure 6** is a more detailed flowchart of operations of a middle-tier server according to further embodiments of the present invention;

**Figure 7** is a flowchart illustrating operations of a back-end server according to embodiments of the present invention;

5        **Figure 8** is a flowchart of operations of a middle-tier server according to alternative embodiments of the present invention; and

**Figure 9** is a flowchart illustrating operations of a back-end server according to alternative embodiments of the present invention.

10                                **Detailed Description of the Invention**

The present invention now will be described more fully hereinafter with reference to the accompanying drawings, in which illustrative embodiments of the invention are shown. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art.

As will be appreciated by one of skill in the art, the present invention may be embodied as a method, data processing system, or computer program product.

20    Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects all generally referred to herein as a "circuit" or "module." Furthermore, the present invention may take the form of a computer program product on a computer-usable storage medium having computer-usable

25    program code embodied in the medium. Any suitable computer readable medium may be utilized including hard disks, CD-ROMs, optical storage devices, a transmission media such as those supporting the Internet or an intranet, or magnetic storage devices.

Computer program code for carrying out operations of the present invention

30    may be written in an object oriented programming language such as Java®, Smalltalk or C++. However, the computer program code for carrying out

operations of the present invention may also be written in conventional procedural programming languages, such as the "C" programming language. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer. In the latter scenario, the remote computer may be connected to the user's computer through a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

10           The present invention is described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for  
15           implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

          These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function/act specified in the flowchart and/or block diagram block or blocks.

          The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the

computer or other programmable apparatus provide steps for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

Various embodiments of the present invention will now be described with reference to the figures. As will be appreciated by those of skill in the art in light  
5 of the present disclosure, while embodiments of the present invention are described primarily with reference to PKI, embodiments of the present invention may also provide for authentication of messages utilizing other security protocols which utilize a nonce.

**Figure 1A** illustrates a network configuration where multiple back-end  
10 servers **20**, **22** and **24** are accessed by a middle-tier server **14** in which embodiments of the present invention may be incorporated. As seen in **Figure 1A**, a principal may use a client data processing system **10** to access a server, such as the back-end servers **20**, **22** and **24**. Thus, the client **10** and the back-end servers **20**, **22** and **24** are endpoints respectively for requesting and providing a work unit.  
15 The servers between these endpoints may be referred to as "middle-tier servers." In the exemplary system illustrated in **Figure 1A**, the client **10** communicates with one or more middle-tier servers **14** to access the back-end servers **20**, **22** and **24**. The exemplary single middle-tier server **14** is delegated authority by the client **10** to act on its behalf in accessing the back-end servers **20**, **22** and **24** through the use  
20 of a common nonce as described herein.

As is further seen in **Figure 1A**, the middle-tier server **14** requests a pre-nonce contribution from each of the back-end servers **20**, **22** and **24**. In response to these requests, the back-end servers **20**, **22** and **24** send pre-nonce contributions to the middle-tier server **14**. The middle-tier server combines the pre-nonce  
25 contributions from the back-end servers **20**, **22** and **24** and creates a common nonce. The nonce is provided to the client **10** for use by the client **10** in accessing the back-end servers **20**, **22** and **24**. To authenticate itself, the client **10** signs the nonce with its digital signature and returns the signed nonce to the middle-tier server which provides the signed nonce to the back-end servers **20**, **22** and **24** when  
30 access of these servers on behalf of the client **10** is desired.



As illustrated in **Figure 1A**, the signed nonce may be provided to the back-end servers **20**, **22** and **24** as part of an authenticated request packet which may include, for example, the signed nonce, the request from the client, the pre-nonce token and/or the client certificate. Furthermore, as illustrated by the dashed lines in

5 **Figure 1A**, the back-end server **20** may, optionally, forward the authenticated request packet to other servers, such as the back-end server **24**. Thus, further downstream chaining of the delegated authority may also be provided.

**Figure 1B** illustrates an alternative embodiment of the present invention where the common nonce is provided by a trusted third party **18** such that the  
10 common nonce may be authenticated as being from the trusted third party **18** by each of the middle-tier server **14** and the back-end servers **20**, **22** and **24**. Such a common nonce may be generated by the trusted third party **18**. The nonce may be, but is not required to be, generated based on input from the back-end servers **20**, **22** and **24**. The nonce from the trusted third party **18** is provided to the client **10** for  
15 use by the client **10** in accessing the back-end servers **20**, **22** and **24**. To authenticate itself, the client **10** signs the nonce with its digital signature and returns the signed nonce to the middle-tier server which provides the signed nonce to the back-end servers **20**, **22** and **24** when access to these servers on behalf of the client **10** is desired.

20 As with the embodiments illustrated in **Figure 1A**, in the embodiments illustrated in **Figure 1B** the signed nonce may be provided to the back-end servers **20**, **22** and **24** as part of an authenticated request packet which may include, for example, the signed nonce, the request from the client, the pre-nonce token and/or the client certificate. Furthermore, as illustrated by the dashed lines in **Figure 1B**,  
25 the back-end server **20** may, optionally, forward the authenticated request packet to other servers, such as the back-end server **24**. Thus, in the embodiments illustrated in **Figure 1B** further downstream chaining of the delegated authority may also be provided.

While **Figures 1A** and **1B** are illustrated as having a single client **10** and a  
30 single middle-tier server **14** and three back-end servers **20**, **22** and **24**, other combinations, including multiple clients and/or servers, may also be provided.

Thus, the present invention should not be construed as limited to the configurations of **Figures 1A** and **1B** but is intended to cover all configurations capable of carrying out the operations described herein.

**Figure 2** illustrates an exemplary embodiment of a data processing system **230** suitable for providing a client, a middle tier server, a trusted third party and/or a back-end server in accordance with embodiments of the present invention. The data processing system **230** typically includes input device(s) **232** such as a keyboard or keypad, a display **234**, and a memory **236** that communicate with a processor **238**. The data processing system **230** may further include a speaker **244**, and an I/O data port(s) **246** that also communicate with the processor **238**. The I/O data ports **246** can be used to transfer information between the data processing system **230** and another computer system or a network. These components may be conventional components such as those used in many conventional data processing systems which may be configured to operate as described herein.

**Figure 3** is a block diagram of embodiments of data processing systems that illustrates systems, methods, and computer program products in accordance with embodiments of the present invention. The processor **238** communicates with the memory **236** via an address/data bus **348**. The processor **238** can be any commercially available or custom microprocessor. The memory **236** is representative of the overall hierarchy of memory devices containing the software and data used to implement the functionality of the data processing system **230**. The memory **236** can include, but is not limited to, the following types of devices: cache, ROM, PROM, EPROM, EEPROM, flash memory, SRAM, and DRAM.

As shown in **Figure 3**, the memory **236** may include several categories of software and data used in the data processing system **230**: the operating system **352**; the application programs **354**; the input/output (I/O) device drivers **358**; and the data **356**. As will be appreciated by those of skill in the art, the operating system **352** may be any operating system suitable for use with a data processing system, such as OS/2, AIX or System390 from International Business Machines Corporation, Armonk, NY, Windows95, Windows98 or Windows2000 from Microsoft Corporation, Redmond, WA, Unix or Linux configured to support an

TCP/IP-based protocol connection. The I/O device drivers 358 typically include software routines accessed through the operating system 352 by the application programs 354 to communicate with devices such as the I/O data port(s) 246, the data storage 235 and certain memory 236 components. The application programs 5 354 are illustrative of the programs that implement the various features of the data processing system 230 and preferably include at least one application which supports operations according to embodiments of the present invention. Finally, the data 356 represents the static and dynamic data used by the application programs 354, the operating system 352, the I/O device drivers 358, and other 10 software programs that may reside in the memory 236.

As is further seen in **Figure 3**, the application programs 354 may include a nonce processing module 360. The nonce processing module 360 may carry out the operations described herein for management, generation and/or use of common nonces for accessing multiple servers. The data portion 356 of memory 236, as 15 shown in the embodiments of **Figure 3**, may include nonce values 364 and, in certain embodiments, pre-nonce values 362. The nonce values 364 and the pre-nonce values 362 may be utilized by the nonce processing module 360 as described herein for authentication of a client to multiple servers.

While the present invention is illustrated, for example, with reference to the 20 nonce processing module 360 being an application program in **Figure 3**, as will be appreciated by those of skill in the art, other configurations may also be utilized while still benefitting from the teachings of the present invention. For example, the nonce processing module 360 may also be incorporated into the operating system 352, the I/O device drivers 358 or other such logical division of the data 25 processing system 230. Thus, the present invention should not be construed as limited to the configuration of **Figure 3** but is intended to encompass any configuration capable of carrying out the operations described herein.

Operations according to embodiments of the present invention, including operations of the middle-tier server 14 and the back-end servers 20, 22 and 24 will 30 now be described with reference to the exemplary network diagram of **Figures 1A** and **1B** and the flowcharts of **Figures 4** through **9**. However, as will be

appreciated by those of skill in the art, the networks illustrated in **Figures 1A** and **1B** are exemplary only. Embodiments of the present invention may be utilized in other network configurations.

As seen in **Figure 4**, a middle-tier server **14** according to embodiments of the present invention provides a common nonce to the client **10** which may be used to access to each of the plurality of back-end servers **20, 22** and **24** (block **400**). Such a common nonce may be generated by the middle-tier server **14** (**Figure 1A**) or provided by a trusted third party **18** (**Figure 1B**) or other server. In either case, the middle-tier server **14** sends the common nonce to the client **10** requesting access to one or more of the back-end servers **20, 22** and **24** (block **405**). The middle-tier server **14** receives back from the client **10** the common nonce signed with the digital signature of the client **10** (block **410**). The middle-tier server **14** uses the signed common nonce to access the one or ones of the back-end servers **20, 22** and **24** on behalf of the client **10** by providing the signed common nonce to the back-end servers for authentication (block **415**).

**Figures 5** through **7** illustrate operations carried out by the middle-tier server and/or the back-end servers **20, 22** and **24** in the embodiments illustrated in **Figure 1A** where the common nonce is generated by a middle-tier server. Alternatively, the operations could be carried out by the trusted third part **18** of **Figure 1B** or other server to generate the common nonce which is provided to the middle-tier server **14**.

**Figure 5** illustrates operations for the generation of a common nonce according to embodiments of the present invention. As seen in **Figure 5**, a request is sent to the back-end servers **20, 22** and **24** to request a pre-nonce contribution from the back-end servers **20, 22** and **24** (block **500**). The request may, for example, include an identification of the requestor, such as the middle-tier server **14**, a timestamp and/or a random number associated with the requestor. The random number may, like a nonce, be a 60 bit value. The request may, optionally, be provided over an encrypted link and may utilize authentication by the requestor and/or the receiving server. Furthermore, the request may be provided to one or

more of the back-end servers which may obtain and provide pre-nonce contributions for other of the back-end servers.

In response to the request, pre-nonce contributions are received from the back-end servers 20, 22 and 24 (block 505). As described above, the pre-nonce contributions may be received directly from each server or from other servers acting on behalf of the back-end servers. Thus, a pre-nonce contribution is received for each of the back-end servers. Such pre-nonce contributions may, for example, include an identification of the back-end server associated with the pre-nonce contribution and/or a random number associated with the back-end server. The random number may, like a nonce, be a 60 bit value. Optionally, the pre-nonce contributions may include a signature associated with the server. As described below, the back-end server could also maintain a list of random numbers it has provided as a pre-nonce contribution and associate an expiration time with corresponding numbers.

When pre-nonce contributions have been received from all of the back-end servers associated with the common nonce being generated (block 510), a pre-nonce token is created based on all of the received pre-nonce contributions (block 515). Such a pre-nonce token may, for example, be created by concatenating the received pre-nonce contributions. Preferably, such concatenation is done in a predefined manner. The common nonce is generated from the pre-nonce token by, for example, reducing the pre-nonce token to a 60 bit value (block 520). Such a reduction may, for example, be provided by a one-way hash function, such as the SHA or MD5 hash functions, which hash the n-bit pre-nonce token to the 60 bit nonce length of PKI to provide the common nonce.

Optionally, a transaction identification could also be obtained from a trusted back-end server. The transaction identification could be associated with the common nonce and even incorporated into the common nonce generation procedure through the concatenation process. The transaction identification could be utilized to provide an audit trail for tracking usage of the common nonce.

**Figure 6** illustrates operations for use of the common nonce for impersonation of the client 10 according to embodiments of the present invention.

As seen in **Figure 6**, the middle-tier server **14** receives a request from the client **10** for a transaction with one or more of the back-end servers **20, 22** and **24** (block **600**). The middle-tier server **14** sends the common nonce to the client **10** (block **605**) which signs the common nonce with its digital signature and returns the  
5 signed common nonce as its signature to the middle-tier server **14**. The middle-tier server receives the signed common nonce (block **610**) and sends the pre-nonce token, client certificate, common nonce and client signature to the back-end server to authenticate the client to the back-end server (block **615**). Additionally, as described above, optionally, the middle-tier server **14** may track use of the common  
10 nonce (block **620**), for example, by use of the transaction identification and creation of a log file or other such tracking mechanism.

**Figure 7** illustrates operations of a back-end server upon receipt of the information from the middle-tier server **14**. As seen in **Figure 7**, the back-end server receives the information from the middle-tier server **14** (block **700**) and  
15 determines if the certificate received from the middle-tier server **14** is from a trusted client (block **705**). Such a determination may be made using conventional PKI techniques. If the certificate is not from a trusted client (block **705**), the information is rejected as not authenticated (block **740**). If the certificate is from a trusted client (block **705**), it is determined if the signature corresponds to the  
20 certificate signature (block **710**). Such a determination may be made using conventional PKI techniques for verification of a signature based on a ticket. If the signature does not correspond to the certificate signature (block **710**), the information is rejected as not authenticated (block **740**).

If the signature corresponds to the certificate signature (block **710**), the pre-  
25 nonce token is hashed using the same hash technique utilized by the middle-tier server (block **715**) and the results are compared to the common nonce (block **720**). If the results are not the same as the common nonce (block **720**), the information is rejected as not authenticated (block **740**). If the results are the same as the common nonce (block **720**), the pre-nonce token is evaluated to determine if it  
30 contains the random number provided by the back-end server as part of its pre-nonce contribution (block **725**). While not illustrated in **Figure 7**, if the random

number is optionally signed as described above, the digital signature may be evaluated to determine if it is the digital signature of the back-end server. If the signature does not correspond to that of the back-end server the information would be rejected as not authenticated. Returning to **Figure 7**, if the random number  
5 provided by the back-end server is not present in the pre-nonce token (block **725**), the information is rejected as not authenticated (block **740**).

Optionally, if the random number is present in the pre-nonce token (block **725**) and the random numbers provided by the back-end server have expirations which are tracked by the back-end server, it may be determined if the random  
10 number has expired (block **730**). If the random number has expired (block **730**), the information is rejected as not authenticated (block **740**). If the random number has not expired (block **730**), the information is accepted as an authenticated client impersonated by the middle-tier server **14** (block **735**).

**Figures 8 and 9** illustrate operations according to alternative embodiments  
15 of the present invention where the common nonce is obtained from the trusted third party **18**. As seen in **Figure 8**, the common nonce may be generated by sending a request from the middle-tier server **14** to the trusted third party **18** (block **800**). The common nonce and signature of the trusted third party are received (block **805**) and it is determined if the signature is authentic (block **810**). If the signature  
20 is not authentic (block **810**), the common nonce is rejected (block **820**). If the signature is authentic (block **810**), the common nonce is accepted (block **815**). The common nonce may be used by the middle-tier server **14** to impersonate the client **10** as described with reference to **Figure 6** with appropriate revision to the information provided to the back-end server so as to provide the signature of the  
25 client and the signature of the trusted third party to the back-end server.

**Figure 9** illustrates operations of the back-end server according to alternative embodiments of the present invention. The back-end server receives the information from the middle-tier server **14** (block **900**) and determines if the certificate received from the middle-tier server **14** is from a trusted client (block  
30 **905**). Such a determination may be made using conventional PKI techniques. If the certificate is not from a trusted client (block **905**), the information is rejected as

not authenticated (block 930). If the certificate is from a trusted client (block 905), it is determined if the signature corresponds to the client certificate signature (block 910). Such a determination may be made using conventional PKI techniques for verification of a signature based on a ticket. If the signature does not correspond to the certificate signature (block 910), the information is rejected as not authenticated (block 930).

If the signature corresponds to the client certificate signature (block 910), it is determined if the third party certificate is from a trusted third party (block 915). If not, the information is rejected as not authenticated (block 930). If the third party certificate is from a trusted third party (block 915), the signature of the common nonce is evaluated to determine if it corresponds to the signature from the trusted third party certificate (block 920). If not, the information is rejected as not authenticated (block 930). If the signature of the common nonce corresponds to the signature of the trusted third party certificate (block 920), the information is accepted as an authenticated client impersonated by the middle-tier server (block 925).

While the present invention has been described, in part, with reference to the back-end servers 20, 22 and 24, as will be appreciated by those of skill in the art in light of the present disclosure, the back-end servers 20, 22 and 24 are illustrative of resource managers in general. Accordingly, embodiments of the present invention may be utilized to securely access resource managers of differing types. The resource managers may include, for example, security policies for carrying out some or all of the operations described above for authentication of requests to access a resource managed by the resource manager.

The flowcharts and block diagrams of **Figures 1A, 1B and 2 through 9** illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products for authentication according to various embodiments of the present invention. In this regard, each block in the flow charts or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative



implementations, the functions noted in the blocks may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be understood that each block of the block diagrams and/or flowchart illustrations, and combinations of blocks in the block diagrams and/or flowchart illustrations, can be implemented by special purpose hardware-based systems which perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

10 Furthermore, in the drawings, certain operations/acts and or blocks which have been indicated as optional are shown in dashed lines. The presence or lack of such a dashed line shall not, however, be construed as requiring any of the elements of the figures other than those necessary to provide the functions, operations and/or acts specified in the claims set forth below.

15 In the drawings and specification, there have been disclosed typical illustrative embodiments of the invention and, although specific terms are employed, they are used in a generic and descriptive sense only and not for purposes of limitation, the scope of the invention being set forth in the following claims.

20